Engineering adaptive user interfaces for enterprise applications

Conference or Workshop Item

Engineering Adaptive User Interfaces for Enterprise Applications

Pierre A. Akiki
Computing Department, The Open University
Walton Hall, Milton Keynes, United Kingdom
pierre.akiki@open.ac.uk

ABSTRACT
The user interface (UI) layer is considered an important component in software applications since it links the users to the software's functionality. Enterprise applications such as enterprise resource planning and customer relationship management systems have very complex UIs that are used by users with diverse needs in terms of the required features and layout preferences. The inability to cater for the variety of user needs diminishes the usability of these applications. One way to cater for those needs is through adaptive UIs. Some enterprise software providers offer mechanisms for tailoring UIs based on the variable user needs, yet those are not generic enough to be used with other applications and require maintaining multiple UI copies manually. A generic platform based on a model-driven approach could be more reusable since operating on the model level makes it technology independent. The main objective of this research is devising a generic, scalable, and extensible platform for building adaptive enterprise application UIs based on a runtime model-driven approach. This platform primarily targets UI simplification, which we defined as a mechanism for increasing usability through adaptive behavior by providing users with a minimal feature-set and an optimal layout based on the context-of-use. This paper provides an overview of the research questions and methodology, the results that were achieved so far, and the remaining work.

Author Keywords
Adaptive user interfaces; Simplification;
Enterprise applications; Model-driven engineering

ACM Classification Keywords
[Software Engineering]: D.2.11 Software Architectures - Domain-specific architectures; D.2.2 Design Tools and Techniques - User interfaces; [Information Interfaces and Presentation]: H.5.2 User Interfaces – User-centered design

INTRODUCTION
Enterprise applications (e.g., enterprise resource planning, customer relationship management, etc.) generally serve various purposes in an enterprise's functional business areas such as: Accounting, finance, marketing, inventory, etc. The heavy dependence on these applications drives business owners to ask for UIs that maximize employee efficiency and effectiveness. Yet, as existing research [22] and industry reports [17] have shown, enterprise applications are regarded as lacking in usability and incapable of catering for the variety in user needs. Adaptive behavior has been suggested as a means for enhancing usability [6] and some works particularly suggested applying it to enterprise application UIs [22]. Also, it has been used for tailoring UIs based on several aspects such as: "Accessibility" [14], "Culture" [20], "Natural Context" [7], etc.

A model-driven development approach could form a basis for devising adaptive UIs due to the ability of representing UIs on multiple levels of abstraction that can be loaded and adapted at runtime. The CAMELEON reference framework [9] represents UIs on multiple levels of abstraction: (1) *Tasks Models* can be represented as ConcurTaskTrees [19] and *Domain Models* as UML class diagrams, (2) *Abstract User Interface* (AUI), represents the UI independent of any modality (e.g., Graphical, Voice, etc.), (3) *Concrete User Interface* (CUI), represents the UI as concrete widgets (e.g., Buttons, Labels, etc.), and (4) *Final User Interface* (FUI), is the running UI rendered in a presentation technology.

The primary objective of this research is devising a generic, scalable, and extensible platform for building adaptive enterprise application UIs based on a runtime model-driven approach. The main target of this platform would be UI simplification, which we defined [2] as a mechanism for increasing usability through adaptive behavior by providing users with a minimal feature-set and an optimal layout based on the context-of-use (user, platform, environment).

The remainder of the paper is organized as follows: The next section states and explains the proposed research questions. Then, the related work is briefly discussed and evaluated in the context of the research questions. Later, the research methodology is explained. Afterwards, the results that the research has yielded so far are presented. Finally, the conclusions are given and the remaining work is stated.

RESEARCH QUESTIONS
This work will answer the following main research question from which three sub-questions were derived:

Authors' Version
EICS'13, London, United Kingdom, *Forthcoming*

How can adaptive UI behavior be leveraged for simplifying enterprise applications in order to increase their usability?

Software companies attempt to develop user interfaces that are capable of accommodating the vast majority of an application's target users. Due to the differences in end-user needs, when user interfaces are concerned one does not fit all. For example, if a UI is developed with full functionality it might be over-bloated for basic users. Yet, removing functionality would prevent advanced users from fulfilling their tasks. Also, certain CUI related choices (e.g., type of widgets, layout grouping, etc.) might allow some users to perform their tasks more efficiently in certain contexts-of-use (e.g., a different widget grouping for a mobile phone UI than for a desktop UI, novice users could have widget preferences such as radios over combos, etc.). Another, scenario involves daily tasks that require the use of functionality scattered across multiple UIs. Monitoring user behavior could allow this functionality to be grouped under one UI to make the fulfillment of daily tasks more efficient.

Identifying the various user needs, especially for generic enterprise applications, would be difficult to do at design time. Furthermore, developing and maintaining multiple editions of the same UI is costly especially for enterprise applications comprising thousands of user interfaces. The simplification theme targeted in this research is meant to address the existing variety in the needs of enterprise users by leveraging adaptive user interfaces. The following sub-questions elaborate more on the research specifics.

1. *What is an effective way to automatically simplify individual enterprise application user interfaces based on each end-user's needs?*
2. *What is an effective way to compose new user interfaces at runtime from existing ones based on user behavior?*
3. *What will be the impact of the devised simplification mechanism on the end-users' satisfaction and efficiency?*

RELATED WORK

Based on the previously presented research questions, this section discusses the related work in terms of the ability to:

- Minimize a user interface's feature-set and optimize its layout at runtime
- Decompose existing user interfaces into smaller parts at runtime and use those parts to recompose new UIs

Several existing works discuss adapting the feature-set of UIs such as: "Multi-layered UI" [21], "training wheels UI" [10], and "two-interface design" [18]. Yet, these works are theoretical and there is still a need for a tool supported solution that allows developers to minimize a UI's feature-set in practice at runtime based on the users' needs.

Other works use different approaches to target layout adaptation. The Comet [8] is introduced as a set of widgets that support UI plasticity but only target the adaptation of individual widgets and not the entire layout. Supple [14] is a system capable of generating UIs adapted to each user's motor abilities by treating UI generation as an optimization problem. Yet, Supple does not support the various possible levels of abstraction thereby preventing designer input from being made at the CUI level making it difficult to adopt for enterprise applications. Another adaptation approach [5] defines content personalization at design-time, which is stated to be a major limitation. MASP [7] targets ubiquitous UIs in smart environments and promotes runtime modeling of UIs. MASP relies on code for devising the UI and uses a box-based layouting tool to segment the UI for runtime manipulation. This technique does not make it possible to simplify the UI at the widget level since the manipulation is done on the segments that group multiple widgets. It also does not allow new UIs to be created at runtime since the adaptations expect a code-based UI as input.

Graceful degradation is used as a method for supporting UIs on multiple devices [13] and could be used for decomposing/recomposing UIs. Yet, this method's main limitation lies in its design-time application that relies on designer annotations hence it would not work when the adaptations are only known at runtime. An interesting approach would be to combine annotations with automated procedures based on user behavior. Another approach called (de)composition seems to complement some aspects of the graceful degradation process [16]. It aims towards supporting reusability at a high level design without the need for applying constant copy and paste operations. The authors mention the applicability of (de)composition both at design/run-time but all the given examples were restricted to design-time. Decomposing/Composing UIs at runtime would also require adapting the functionality behind the UI.

RESEARCH METHODOLOGY

Easterbrook et al. [11] differentiate between "*knowledge*" and "*design*" research questions. They note that knowledge questions focus on "*the way the world is*", whereas design questions focus on establishing "*better ways to do software engineering*". Empirical research is usually the path chosen by researchers posing knowledge questions as opposed to an engineering approach taken for design questions.

This research follows an engineering approach containing a mixture of both design and knowledge questions. The design questions aim towards coming up with an effective technique for developing enterprise UIs with simplification capabilities based on existing research work. On the other hand, the knowledge question aims towards answering how this technique would perform in a practical scenario.

Several engineering techniques will be employed in this research to answer sub-questions 1 and 2. The proposed techniques include modeling, implementing support tools and prototypes, and conducting performance evaluations.

Surveys will be used for the preliminary investigations whereas lab based usability studies will be conducted for confirmatory validation purposes to answer sub-question 3.

RESULTS

This section discusses the parts of the research that have been accomplished so far.

CEDAR Architecture

The CEDAR architecture [1], illustrated in Figure 1, serves as a reference for devising adaptive model-driven enterprise application UIs. This architecture is based on the: (1) Three Layer Architecture [15] (*Adaptive System Layering*), (2) CAMELEON reference framework [9] (*UI Abstraction*), and (3) Model-View-Controller paradigm (*Implementation*). CEDAR promotes the use of interpreted runtime models instead of code generation for providing more flexibility in performing advanced UI adaptations at runtime. A practical implementation [1] based on CEDAR showed that runtime UI rendering does not negatively impact performance. A major part of CEDAR has been implemented to support our UI simplification mechanism described in the next section.

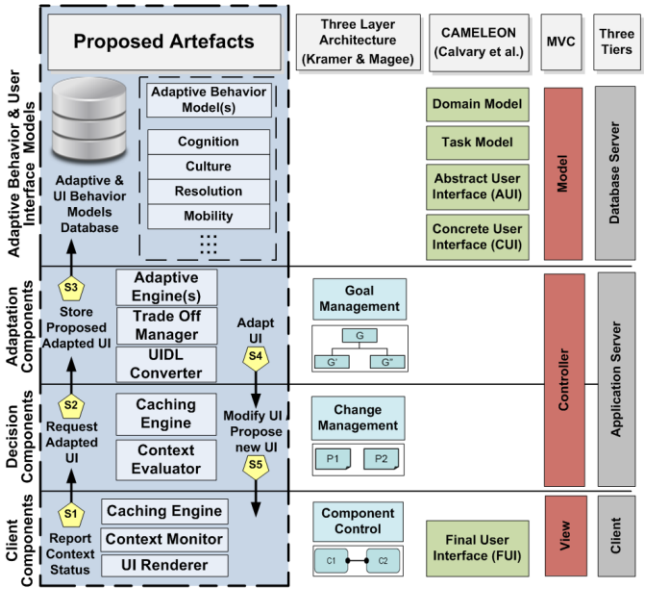
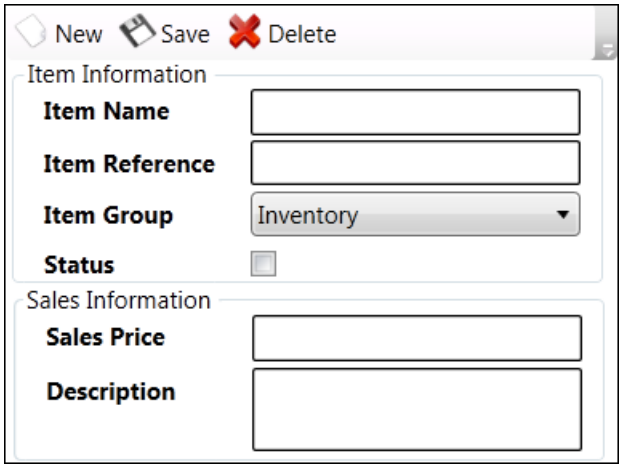


Figure 1: The CEDAR Architecture

Role-Based UI Simplification (RBUIS)

Role-Based UI Simplification (RBUIS) [2] is a mechanism that merges role-based access control (RBAC) [12] with adaptive behavior for simplifying UIs. In RBUIS, *roles* are divided into groups representing the aspects based on which the UI will be simplified such as computer literacy, job title, etc. RBUIS supports *feature-set minimization* by assigning roles to task models for providing users with a minimal feature-set based on the context-of-use. The assignment could be done by I.T. personnel but there is also a potential for engaging end-users in the process [3]. *Layout optimization* is supported by assigning roles to workflows that represent adaptive UI behavior visually and through code and can be applied on CUI models. Furthermore, RBUIS promotes user feedback for refining the adaptation operations. Hence, users are allowed to reverse feature-set minimizations and layout optimizations, and to choose possible alternative layout optimizations. A user-study [2] showed that applying RBUIS enhances the usability of complex user interfaces.

(a) Initial Item Maintenance UI



(b) Simplified Item Maintenance UI

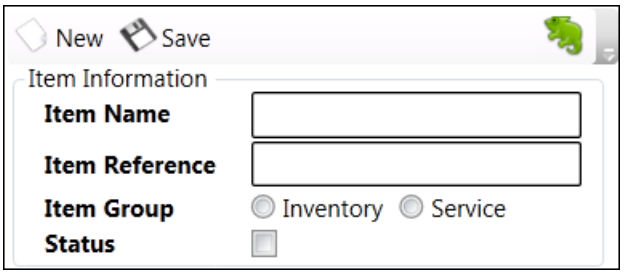


Figure 2: User Interface Simplification with RBUIS

The example illustrated in Figure 2 demonstrates how RBUIS can be applied to simplify UIs by minimizing the feature-set (sales information and delete button are removed in this case) and optimizing the layout (combo-boxes are substituted with radio-buttons in this case). Additionally, the example shows a chameleon icon in the corner of the simplified UI (Figure 2 – b). This icon allows users to view a list of adaptations on which they can provide feedback. The change between versions (a) and (b) is based on the set of roles representing different aspects such as computer literacy, job title, etc. When an enterprise user logs into the system and activates a UI, the version that is loaded on the screen is dynamically adapted according to the roles that have been assigned to the session's user identifier.

Cedar Studio

The *Cedar Studio* IDE [4] provides tool support for building enterprise applications based on the CEDAR architecture. *Cedar Studio* allows developers and I.T. personnel to apply RBUIS using a set of visual design and code editing tools that support the creation of UI models and adaptive behavior. Automatic generation between the levels of abstraction (Task, AUI, and CUI) is supported with the possibility to make manual changes at any level. The CUI designer of *Cedar Studio* is shown in Figure 3.

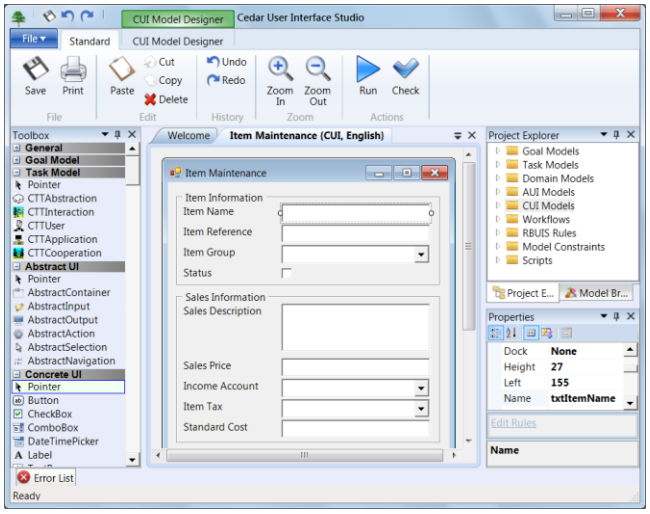


Figure 3: The Cedar Studio IDE

CONCLUSIONS AND REMAINING WORK

This paper presented an overview of an ongoing PhD work on simplifying enterprise application user interfaces through engineering adaptive behavior. The proposed research questions and methodology were explained and the results obtained so far were presented.

In order to fully answer the research questions some work still has to be done. A technique complementary to RBUIS will be proposed to answer the second question on composing new UIs at runtime by monitoring user behavior. This technique will provide the ability to combine features from multiple UIs into a new UI to make it easier to accomplish tasks that require partial features from different UIs. This process has to take into consideration both the layout and the code-behind in order to maintain the UI's functionality. A comprehensive performance study will be conducted to test the entire simplification technique in an industrial scenario. Additionally, more lab studies will be conducted to test the usability of the produced outcome using several example UIs from existing enterprise applications.

ACKNOWLEDGMENTS

This PhD is funded through a three year studentship granted by the Computing Department at The Open University U.K.

REFERENCES

1. Akiki, P.A., Bandara, A.K., and Yu, Y. Using Interpreted Runtime Models for Devising Adaptive User Interfaces of Enterprise Applications. ICEIS'12, SciTePress (2012), 72-77.
2. Akiki, P.A., Bandara, A.K., and Yu, Y. RBUIS: Simplifying Enterprise Application User Interfaces through Engineering Role-Based Adaptive Behavior. EICS'13, ACM (2013), *Forthcoming*.
3. Akiki, P.A., Bandara, A.K., and Yu, Y. Crowdsourcing User Interface Adaptations for Minimizing the Bloat in Enterprise Applications. EICS'13, ACM (2013), *Forthcoming*.
4. Akiki, P.A., Bandara, A.K., and Yu, Y. Cedar Studio: An IDE Supporting Adaptive Model-Driven User Interfaces for Enterprise Applications. EICS'13, ACM (2013), *Forthcoming*.
5. Bacha, F., Oliveira, K., and Abed, M. A Model Driven Architecture Approach for User Interface Generation Focused on Content Personalization. RCIS'11, IEEE (2011), 1-6.
6. Benyon, D. Adaptive systems: a solution to usability problems. User Modeling and User-Adapted Interaction 3, 1 Springer (1993), 65-87.
7. Blumendorf, M., Lehmann, G., and Albayrak, S. Bridging Models and Systems at Runtime to Build Adaptive User Interfaces. EICS'10, ACM (2010), 9-18.
8. Calvary, G., Coutaz, J., Dâassi, O., Balme, L., and Demeure, A. Towards a New Generation of Widgets for Supporting Software Plasticity: The "Comet". Eng. HCI and Interactive Systems. Springer (2005), 306-324.
9. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3, Elsevier (2003), 289-308.
10. Carroll, J.M. and Carrithers, C. Training Wheels in a User Interface. CACM 27, 8, ACM (1984), 800-806.
11. Easterbrook, S., Singer, J., Storey, M.-A., and Damian, D. Selecting Empirical Methods for Software Engineering Research. Guide to Advanced Empirical Software Engineering, Springer (2008), 285-311.
12. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., and Chandramouli, R. Proposed NIST Standard for Role-Based Access Control. TISSEC, ACM (2001), 224-274.
13. Florins, M. and Vanderdonckt, J. Graceful Degradation of User Interfaces as a Design Method for Multiplatform Systems. IUI'04, ACM (2004), 140-147.
14. Gajos, K.Z., Weld, D.S., and Wobbrock, J.O. Automatically Generating Personalized User Interfaces with Supple. Artificial Intelligence, Elsevier (2010), 910-950.
15. Kramer, J. and Magee, J. Self-Managed Systems: an Architectural Challenge.FOSE'07, IEEE (2007), 259-268.
16. Lepreux, S., Vanderdonckt, J., and Michotte, B. Visual Design of User Interfaces by (De)Composition. DSV-IS'07, Springer-Verlag (2007), 157-170.
17. Lykkegaard, B. and Elbak, A. IDC - Document at a Glance - LC52T. International Data Corporation (2011).
18. McGrenere, J., Baecker, R.M., and Booth, K.S. An Evaluation of a Multiple Interface Design Solution for Bloated Software. CHI'02, ACM (2002), 164-170.
19. Paterno, F. Model-based Design and Evaluation of Interactive Applications. Springer-Verlag (1999).
20. Reinecke, K. and Bernstein, A. Improving Performance, Perceived Usability, and Aesthetics with Culturally Adaptive User Interfaces. TOCHI 18, ACM (2011), 1-29.
21. Shneiderman, B. Promoting Universal Usability with Multi-Layer Interface Design. CUU'03, ACM (2003), 1-8.
22. Singh, A. and Wesson, J. Evaluation Criteria for Assessing the Usability of ERP Systems. SAICSIT '09, ACM (2009), 87-95.